

# Cricket Ball Prediction Model V2: Unified Heterogeneous Graph Architecture

## Architecture Documentation

December 18, 2025

### Abstract

This document describes Version 2 of the cricket ball prediction model. The model represents **all information as a single heterogeneous graph** where different types of nodes (venue, players, match state, ball history) are connected by typed edges that encode cricket-specific relationships. This enables processing full innings history while maintaining computational efficiency through sparse, structured attention patterns.

## Contents

<b>1 Overview: What is a Heterogeneous Graph?</b>	<b>3</b>
1.1 The Key Idea . . . . .	3
1.2 Why This Matters for Cricket . . . . .	3
<b>2 Node Types: The Building Blocks</b>	<b>4</b>
2.1 Category 1: Global (3 nodes) . . . . .	4
2.2 Category 2: State (5 nodes) . . . . .	4
2.3 Category 3: Actor (7 nodes) . . . . .	5
2.4 Category 4: Dynamics (4 nodes) . . . . .	5
2.5 Category 5: Ball (N nodes) . . . . .	6
2.6 Category 6: Query (1 node) . . . . .	6
2.7 Summary: Total Node Count . . . . .	6
<b>3 Edge Types: The Relationships</b>	<b>7</b>
3.1 Understanding Edge Type Notation . . . . .	7
3.2 Category 1: Hierarchical Edges (3 types) . . . . .	7
3.3 Category 2: Intra-Layer Edges (4 types) . . . . .	7
3.3.1 What does “global $\leftrightarrow$ global” mean? . . . . .	7
3.3.2 The Actor Matchup Graph . . . . .	8
3.4 Category 3: Temporal Edges (6 types) . . . . .	8
3.4.1 Multi-Scale Temporal Architecture . . . . .	8
3.4.2 Same-Bowler and Same-Batsman Edges (with Temporal Decay) . . . . .	9
3.4.3 Same-Over Edges (CAUSAL) . . . . .	9
3.4.4 Same-Matchup Edges (CAUSAL) . . . . .	10
3.5 Category 4: Cross-Domain Edges (4 types) . . . . .	10
3.6 Category 5: Query Edges (2 types) . . . . .	10
3.6.1 Attends Edges . . . . .	10
3.6.2 Drives Edges (NEW) . . . . .	11
3.7 Edge Type Summary . . . . .	11
<b>4 Full Graph Visualization</b>	<b>12</b>

<b>5 Model Architecture</b>	<b>13</b>
5.1 Three-Stage Pipeline . . . . .	13
5.2 Hybrid Readout: Why? . . . . .	13
5.3 Convolution Choices per Edge Type . . . . .	14
<b>6 Training: Focal Loss for Class Imbalance</b>	<b>15</b>
6.1 Training Configuration . . . . .	15
<b>7 Why Full History Now Works</b>	<b>16</b>
7.1 The Quadratic Problem (V1) . . . . .	16
7.2 The Sparse Solution (V2) . . . . .	16
<b>8 Interpretability Benefits</b>	<b>17</b>
<b>9 Summary</b>	<b>17</b>

# 1 Overview: What is a Heterogeneous Graph?

## 1.1 The Key Idea

A **heterogeneous graph** contains multiple types of nodes and multiple types of edges. Unlike a standard graph where all nodes are the same, our cricket graph has:

- **Different node types:** venue, teams, players, match state, momentum, ball history
- **Different edge types:** “conditions”, “matchup”, “precedes”, “same\_bowler”, etc.

Each edge type can have its own neural network (convolution operator), allowing the model to learn relationship-specific patterns.

## 1.2 Why This Matters for Cricket

Cricket prediction requires understanding many different types of information and their relationships:

- **Who** is batting and bowling (player identities)
- **Where** the match is being played (venue characteristics)
- **What** is the current score, required rate, wickets in hand
- **How** the players are performing today (runs, strike rate, economy)
- **What happened** in previous balls (ball-by-ball history)

By encoding all of this in a single graph, the model can learn complex interactions (e.g., “Rohit struggles against left-arm spin at the MCG in the death overs”).

## 2 Node Types: The Building Blocks

The graph contains **21 context nodes** organized into **6 categories**, plus N ball history nodes.

**Important distinction:**

- A **category** is a semantic grouping (e.g., “Actor” = all player-related nodes)
- Individual **nodes** within a category represent specific pieces of information

### 2.1 Category 1: Global (3 nodes)

These nodes represent match-level constants that don’t change during the innings:

Node Name	Features	What it Represents
venue	32-dim embedding	The ground (MCG, Eden Gardens, etc.). Learned embedding captures pitch behavior, boundaries, historical scoring patterns.
batting_team	32-dim embedding	The team currently batting. Captures team-level tendencies.
bowling_team	32-dim embedding	The team currently bowling.

Table 1: Global category: 3 nodes representing match constants

**Example:** At the MCG, India batting vs Australia bowling → venue, batting\_team, bowling\_team nodes initialized with their learned embeddings.

### 2.2 Category 2: State (5 nodes)

These nodes capture the current match situation. They change as the match progresses:

Node Name	Features	What it Represents
score_state	4 features	Current score, wickets, balls, innings
chase_state	7 features	Runs needed, RRR, is_chase, RRR normalized, chase difficulty, balls remaining, wickets remaining (2nd innings)
phase_state	5 features	Powerplay/middle/death phase, over progress, is_first_ball cold-start indicator
time_pressure	3 features	Balls remaining, urgency, is_final_over
wicket_buffer	2 features	Wickets in hand, is_tail indicator

Table 2: State category: 5 nodes representing current match situation

**Example:** Score 150/3 in 15 overs chasing 180 → score\_state=(150, 3, 10.0), chase\_state=(180, 6.0, 30), phase\_state=(0, 1, 0, 0.75, 0).

**Note:** `is_first_ball` in `phase_state` is a cold-start indicator (1.0 if this is the very first ball of the innings, 0.0 otherwise).

### 2.3 Category 3: Actor (7 nodes)

These nodes represent the current players on the field and their performance:

Node Name	Features	What it Represents
<code>striker_identity</code>	Hierarchical embedding	WHO is on strike (player→team→role fallback for unknown players)
<code>striker_state</code>	8 features	HOW they're batting: runs, balls, SR, dot%, boundaries, set indicator, <code>is_debut_ball</code> , <code>balls_since_on_strike</code>
<code>nonstriker_identity</code>	Hierarchical embedding	WHO is at the non-striker's end (with cold-start fallback)
<code>nonstriker_state</code>	7 features	HOW the non-striker has been batting (includes <code>is_debut_ball</code> )
<code>bowler_identity</code>	Hierarchical embedding	WHO is bowling (with cold-start fallback)
<code>bowler_state</code>	6 features	HOW they're bowling: overs, runs, wickets, economy, dot%
<code>partnership</code>	4 features	Partnership runs, balls, run rate, boundaries

Table 3: Actor category: 7 nodes representing current players

**Key insight:** Separating identity from state allows the model to learn “Bumrah is generally accurate” (identity) independently from “Bumrah is bowling well today” (state).

**Note:** `is_debut_ball` in striker/nonstriker state is 1.0 if this is the player’s first ball of the match (cold-start handling).

**Note:** `balls_since_on_strike` in striker\_state captures the “cold restart” effect: 0.0 if the striker faced the previous ball, increasing toward 1.0 if they’ve been at non-striker end for 12+ balls (2 overs). This helps the model understand that a batsman coming back on strike after a period at non-striker is less “in rhythm” than one continuously facing.

### 2.4 Category 4: Dynamics (4 nodes)

These nodes capture recent momentum and pressure patterns:

Node Name	Features	What it Represents
<code>batting_momentum</code>	4 features	Recent scoring rate, boundary%, acceleration
<code>bowling_momentum</code>	4 features	Recent economy, dot%, control
<code>pressure</code>	3 features	Required rate vs actual rate, mounting pressure
<code>dot_pressure</code>	3 features	Consecutive dots, dot ball spiral indicator

Table 4: Dynamics category: 4 nodes representing momentum and pressure

**Why this matters:** Cricket outcomes are heavily influenced by momentum. A batsman who just hit two boundaries is in a different mental state than one who faced 5 dots.

## 2.5 Category 5: Ball (N nodes)

Each historical ball in the innings becomes a node:

Features	Description
17 numeric features	runs, is_wicket, over, ball_in_over, is_boundary, extras (wide, noball, bye, legbye), wicket types (bowled, caught, lbw, run_out, stumped, other), <b>striker_run_out, nonstriker_run_out</b>
64-dim bowler ID	Embedding of who bowled this ball
64-dim batsman ID	Embedding of who faced this ball

Table 5: Ball nodes: one per historical delivery (now with run-out attribution)

**Run-out attribution:** The striker\_run\_out and nonstriker\_run\_out features disambiguate WHO was run out, enabling better risk assessment.

**Total per ball:**  $17 + 64 + 64 = 145$  features  $\rightarrow$  projected to 128-dim.

## 2.6 Category 6: Query (1 node)

A special node used to aggregate all information for the final prediction:

- Initialized with zeros
- After message passing, contains aggregated information from the entire graph
- Final prediction is made from this node's representation

## 2.7 Summary: Total Node Count

Category	Nodes in Category	Role
Global	3	Match constants
State	5	Current situation
Actor	7	Players on field
Dynamics	4	Momentum/pressure
Ball	N (variable)	Full innings history
Query	1	Prediction aggregation
<b>Total</b>	<b>20 + N</b>	

Table 6: Node count by category

### 3 Edge Types: The Relationships

Edges encode **how nodes relate to each other**. Different edge types have different meanings and use different neural network operations.

#### 3.1 Understanding Edge Type Notation

An edge type is written as: `(source_type, relation, target_type)`

For example: `(global, conditions, state)` means “global nodes have a ‘conditions’ relationship with state nodes”.

**Important:** The categories (global, state, etc.) contain multiple nodes. An edge type like `(global, conditions, state)` means edges connect nodes *within* those categories.

#### 3.2 Category 1: Hierarchical Edges (3 types)

These edges encode a **top-down conditioning** structure: higher-level context influences lower-level interpretation.

Edge Type	Connectivity	Meaning
<code>(global, conditions, state)</code>	$3 \rightarrow 5$ (15 edges)	Venue/teams influence how we interpret the score
<code>(state, conditions, actor)</code>	$5 \rightarrow 7$ (35 edges)	Match situation influences how we interpret player performance
<code>(actor, conditions, dynamics)</code>	$7 \rightarrow 4$ (28 edges)	Current players influence how we interpret momentum

Table 7: Hierarchical edges: top-down conditioning

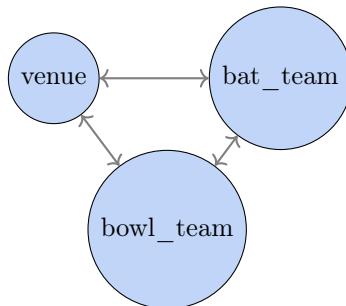
**Concrete example:** 150/3 means different things at the MCG (good) vs Chennai (average). The edge from `venue` to `score_state` allows this contextual interpretation.

#### 3.3 Category 2: Intra-Layer Edges (4 types)

These edges connect nodes **within the same category**, allowing related information to interact.

##### 3.3.1 What does “global $\leftrightarrow$ global” mean?

This means edges between the 3 nodes in the global category:



6 edges total (bidirectional)  
“venue  $\leftrightarrow$  bat\_team”, “venue  $\leftrightarrow$  bowl\_team”, “bat\_team  $\leftrightarrow$  bowl\_team”

Figure 1: Intra-layer edges within the Global category

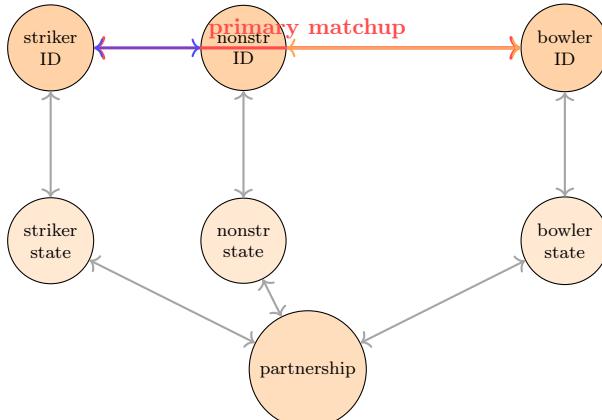
**Why it matters:** India playing at home vs Australia playing away is different from the reverse. The venue-team combination matters.

Edge Type	Edges	What it Captures
(global, relates_to, global)	6	Home/away advantage, venue-team combinations
(state, relates_to, state)	20	Score relates to required rate, phase relates to pressure
(actor, matchup, actor)	22	Striker-bowler confrontation, partnership dynamics
(dynamics, relates_to, dynamics)	12	Batting momentum vs bowling momentum (zero-sum)

Table 8: Intra-layer edges: within-category relationships

### 3.3.2 The Actor Matchup Graph

The (actor, matchup, actor) edges deserve special attention. They encode:



**Red:** Striker vs Bowler (the key prediction)  
**Orange:** Non-striker vs Bowler (run-out risk)  
**Blue:** Striker ↔ Non-striker (partnership chemistry)

Figure 2: Actor matchup edges capture player interactions

## 3.4 Category 3: Temporal Edges (6 types)

These edges encode the **structure of ball-by-ball history**. This is where V2 gains efficiency over transformer attention.

### 3.4.1 Multi-Scale Temporal Architecture

Cricket patterns occur at different time scales:

- **Within-over (1-6 balls):** Immediate pressure, bowler’s current plan
- **Spell window (7-24 balls):** 4-over spell momentum, bowler fatigue patterns
- **Historical (25+ balls):** Phase patterns, earlier wickets’ impact

We use **three separate edge types** for these scales:

Edge Type	Connects	Neural Network
<code>recent_precedes</code>	Balls within 6 deliveries	TransformerConv (fast decay, attention to recency)
<code>medium_precedes</code>	Balls 7-24 apart	TransformerConv (spell-length window)
<code>distant_precedes</code>	Balls 25+ apart (sparse)	SAGEConv (simple mean, sparse connections)

Table 9: Multi-scale temporal edges

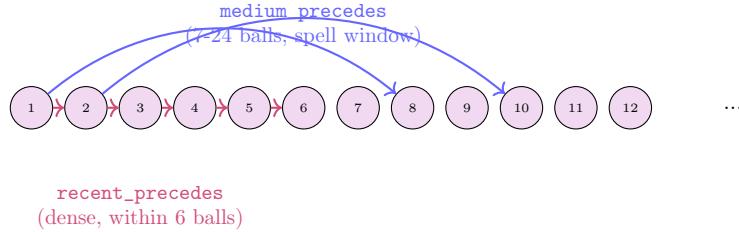
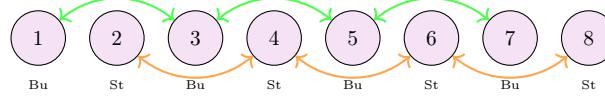


Figure 3: Multi-scale temporal edges (distant not shown for clarity)

### 3.4.2 Same-Bowler and Same-Batsman Edges (with Temporal Decay)

These edges connect balls by the same player, with **temporal decay edge attributes**:



`same_bowler`: Bumrah's deliveries form a connected group  
`same_bowler`: Starc's deliveries form another group

Figure 4: same\_bowler edges group a bowler's spell

#### Temporal decay edge attributes:

- `same_bowler`:  $1.0 - (\text{ball\_distance}/24.0)$  (4-over spell window)
- `same_batsman`:  $1.0 - (\text{ball\_distance}/60.0)$  (10-over innings window)

Recent balls matter more than distant ones. The model uses `TransformerConv` with `edge_dim=1` to learn attention patterns that respect this decay.

**Why this matters:** The model can ask “How has Bumrah been bowling *recently*?” by looking at his deliveries with recency weighting, without needing to attend equally to all balls.

### 3.4.3 Same-Over Edges (CAUSAL)

`same_over` connects balls within the same over. This captures within-over context:

- Same bowler rhythm and plan for the over
- Field placement context
- Batsman reading the bowler

**Critical:** Like same\_matchup, same\_over edges are **CAUSAL** (older → newer only) to prevent train-test distribution shift.

### 3.4.4 Same-Matchup Edges (CAUSAL)

same\_matchup connects balls with the same bowler-batsman combination:

**Critical:** Both same\_over and same\_matchup edges are **UNIDIRECTIONAL** (older → newer). This prevents “future information leakage”:

- During training, the graph contains future balls
- Bidirectional edges would let the model “see” future outcomes
- At inference time, only historical balls exist
- Causal edges ensure the model learns patterns valid at inference

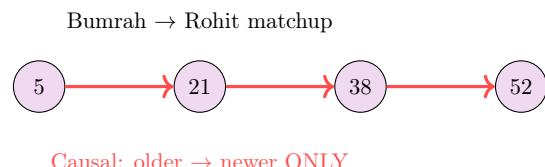


Figure 5: same\_matchup edges are causal to prevent distribution shift

## 3.5 Category 4: Cross-Domain Edges (4 types)

These edges connect ball history to current context nodes:

Edge Type	Meaning	Connectivity
(ball, faced_by, striker_identity)	Balls faced by current striker	Variable
(ball, bowled_by, bowler_identity)	Balls bowled by current bowler	Variable
(ball, partnered_by, nonstriker_identity)	Balls involving current non-striker	Variable
(ball, informs, dynamics)	Recent balls inform momentum	12 → 4

Table 10: Cross-domain edges connect history to context

### Critical: Correct Player Attribution

Cross-domain edges ONLY connect to balls involving the CURRENT players:

- If Kohli faced balls 1-20 then got out, and Sharma is now batting
- Balls 1-20 do NOT connect to `striker_identity` (which is now Sharma)
- Only balls Sharma actually faced will connect

This respects the Z2 symmetry of striker/non-striker swapping.

## 3.6 Category 5: Query Edges (2 types)

### 3.6.1 Attends Edges

The query node receives information from all other nodes:

Edge Type	Connectivity
(* , attends, query)	All 20 context nodes + N balls → query

### 3.6.2 Drives Edges (NEW)

Dynamics nodes **directly drive** prediction through feedback loops:

Edge Type	Meaning
(dynamics, drives, query)	Momentum and pressure directly influence prediction

**Why a separate edge type?**

The **drives** edges capture **causal feedback loops**:

1. **Confidence Spiral:** High batting momentum → more aggressive shots → more runs → higher momentum
2. **Required Rate Pressure:** High pressure → risk-taking → boundaries OR wickets
3. **Dot Ball Spiral:** High dot pressure → desperate shots → wickets → rebuilding

By having learned attention weights on **drives** edges, the model can weight different dynamics signals appropriately.

## 3.7 Edge Type Summary

Category	Edge Types	Purpose
Hierarchical	3	Top-down context conditioning
Intra-layer	4	Within-category interactions
Temporal	7	Ball history (multi-scale + actor grouping + <b>same_over</b> )
Cross-domain	4	Connect history to context
Query	2	Prediction aggregation + dynamics drive
<b>Total</b>	<b>20</b>	

Table 11: 20 edge types organized by purpose (now includes **same\_over** edges)

## 4 Full Graph Visualization

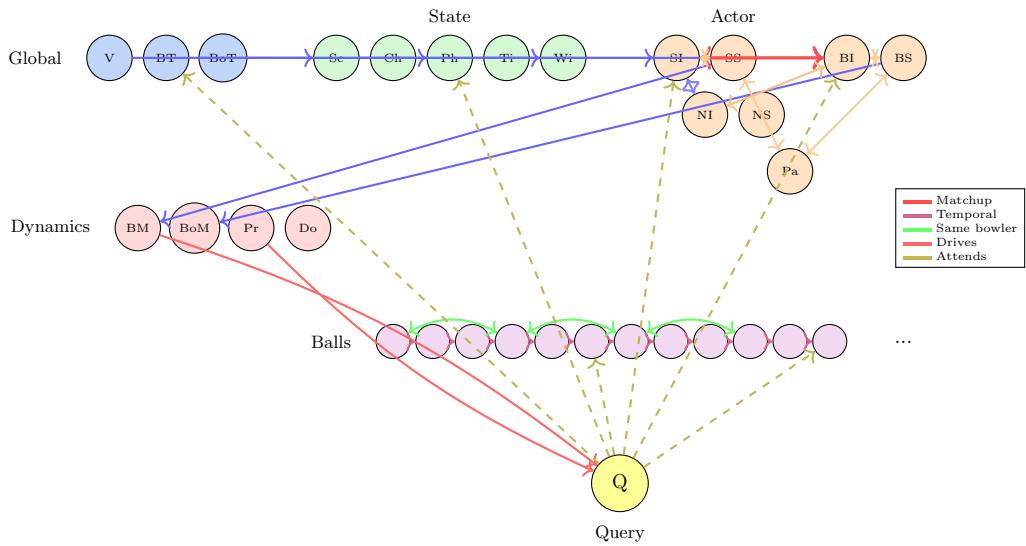


Figure 6: Complete unified graph structure

## 5 Model Architecture

### 5.1 Three-Stage Pipeline

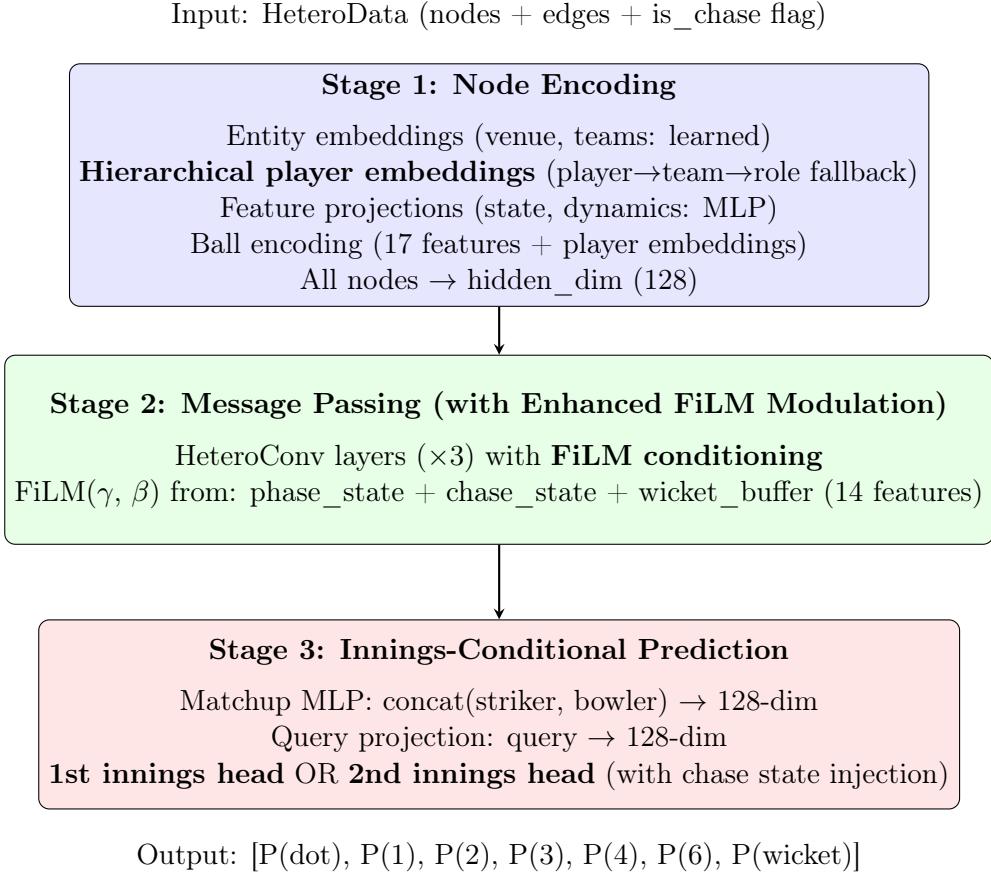


Figure 7: Three-stage model pipeline with FiLM modulation and innings-conditional heads

### 5.2 Hybrid Readout: Why?

Cricket ball prediction is fundamentally an **edge-level task**: the outcome depends on the specific striker-bowler matchup, modulated by context.

The hybrid readout combines:

1. **Matchup interaction:** Direct combination of striker and bowler representations after message passing
2. **Query aggregation:** Global context (venue, phase, momentum) aggregated by the query node

This respects the geometric structure: prediction is at the matchup level, influenced by graph-level context.

### 5.3 Convolution Choices per Edge Type

Edge Type	Convolution	Rationale
Hierarchical (conditions)	GATv2Conv	Learn which context matters
Intra-layer (relates_to)	GATv2Conv	Self-attention for interactions
Actor matchup recent/medium_precedes	GATv2Conv TransformerConv	Learn matchup dynamics Edge features for temporal distance
distant_precedes	SAGEConv	Simple mean for sparse history
<b>same_bowler/batsman</b>	<b>TransformerConv</b>	<b>Temporal decay edge features (4/10-over)</b>
same_matchup (causal)	GATv2Conv	Historical matchup outcomes
<b>same_over (CAUSAL)</b>	<b>TransformerConv</b>	<b>Ball-in-over position edge features</b>
Cross-domain (faced/bowled/partnered)	GATv2Conv	Attention-weighted recency
Dynamics (informs)	<b>GATv2Conv</b>	<b>Attention for important balls</b>
Dynamics (drives)	GATv2Conv	Weight momentum vs pressure
Query (attends)	GATv2Conv	Learn what matters for prediction

Table 12: Edge-type-specific convolution operators (updated with temporal decay and same\_over)

## 6 Training: Focal Loss for Class Imbalance

Cricket outcomes are heavily imbalanced:

- Dots: ~35-40%
- Singles: ~25-30%
- Twos: ~5-8%
- Threes: ~1-2%
- Fours: ~12-15%
- Sixes: ~5-8%
- Wickets: ~5%

Standard cross-entropy loss is dominated by the majority class (dots). We use **Focal Loss**:

$$\mathcal{L}_{focal} = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

where:

- $p_t$  = model's probability for the correct class
- $\gamma = 2.0$  = focusing parameter (higher = more focus on hard examples)
- $\alpha_t$  = optional class weights

**Key insight:**  $(1 - p_t)^\gamma$  down-weights “easy” examples where the model is already confident, focusing learning on hard/rare cases like wickets.

### 6.1 Training Configuration

Parameter	Value
Hidden dimension	128
Number of layers	3
Attention heads	4
Dropout	0.1
Loss	Focal Loss ( $\gamma = 2.0$ )
Optimizer	AdamW (lr=1e-3, wd=0.01)
Scheduler	Cosine annealing
Early stopping	Patience 10 epochs
Estimated parameters	~720K

Table 13: Model and training configuration

## 7 Why Full History Now Works

### 7.1 The Quadratic Problem (V1)

In V1’s Transformer, every ball attends to every other ball:

$$\text{Attention pairs} = \frac{n(n - 1)}{2} = O(n^2)$$

History Length	Attention Pairs
24 balls	276
60 balls	1,770
120 balls (full innings)	7,140

Table 14: Quadratic growth in V1

### 7.2 The Sparse Solution (V2)

In V2, attention only flows along explicit edges. Cricket structure provides natural sparsity:

- Only  $\sim 6$  bowlers per innings  $\rightarrow$  sparse same\_bowler
- Only  $\sim 4\text{-}6$  batsmen  $\rightarrow$  sparse same\_batsman
- Matchups are intersection of above  $\rightarrow$  very sparse
- Multi-scale temporal limits long-range connections

History Length	V1 Attention	V2 Edges	Speedup
24 balls	576	$\sim 150$	4x
60 balls	3,600	$\sim 800$	4.5x
120 balls (full innings)	14,400	$\sim 2,500$	<b>6x</b>

Table 15: Computational comparison: V1 vs V2

**Key insight:** V2 is not just faster—it’s more informative. The 20 context nodes (venue, players, dynamics) provide rich structure that V1 processed separately.

## 8 Interpretability Benefits

The unified graph provides natural interpretability:

1. **Edge attention weights**: Which historical balls mattered most?
2. **Same-bowler attention**: How much did the bowler’s spell inform the prediction?
3. **Matchup attention**: How important was the striker-bowler confrontation?
4. **Dynamics drives attention**: Was momentum or pressure more influential?
5. **Hierarchical attention**: Did venue or match state dominate?

All of these are directly extractable from the GATv2Conv attention weights.

## 9 Summary

The V2 architecture represents cricket prediction as a heterogeneous graph where:

- **21 context nodes** capture venue, teams, match state, players, and dynamics
- **N ball nodes** capture full innings history (17 features including run-out attribution)
- **20 edge types** encode cricket-specific relationships (now includes same\_over)
- **Multi-scale temporal edges** capture patterns at different time scales (6/24/25+ balls)
- **Temporal decay edge attributes** for same\_bowler (4-over) and same\_batsman (10-over)
- **Causal same\_over edges** with **ball-in-over position features** via TransformerConv
- **Causal same\_matchup edges** prevent train-test distribution shift
- **balls\_since\_on\_strike** feature captures striker “cold restart” after non-striker period
- **Hierarchical player embeddings** handle cold-start (unknown players)
- **Enhanced FiLM modulation** conditioned on phase + chase + resource state (14 features)
- **Enhanced chase\_state** with 7 features (RRR, difficulty, resources)
- **GATv2Conv for informs edges** allows attention-weighted momentum aggregation
- **Innings-conditional prediction heads** separate 1st/2nd innings logic (7-dim chase injection)
- **Hybrid readout** combines matchup interaction with global context
- **Focal loss** handles class imbalance

This design follows Geometric Deep Learning principles: the inductive bias matches cricket’s inherent structure, enabling efficient processing of full innings while respecting symmetries (player attribution, temporal causality, phase dynamics).